The Prefabricated Website

Who needs a server anyway?

Martin Holmes and Joseph Takeda
University of Victoria Humanities Computing and Media Centre

Quick summary

The projects: *Endings* and *The Map of Early Modern London*

What is a static site, and why would you build one?

The site build process

Advantages

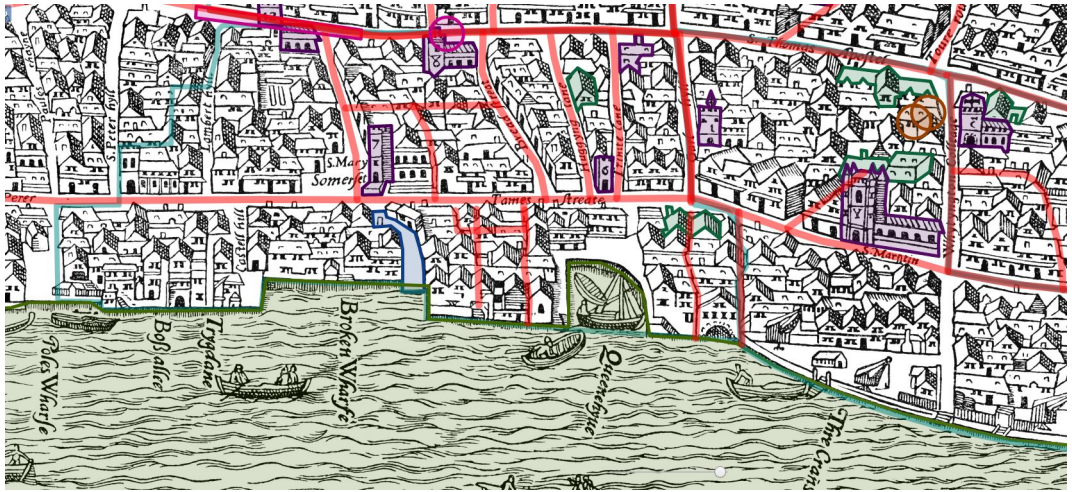
Disadvantages

The Map of Early Modern London

Currently version 6.3 / 6.4, versions dating back to 1999

2,000+ documents, 1,600+ locations, 6,500+ people, 1800+ bibliography items

Gazetteer / encyclopedia / digital edition publication platform



/e Project *Endings*

Endings is a four-year project funded by the Social Sciences and Humanities Research Council (SSHRC) that is creating tools, principles, policies and recommendations for digital scholarship practitioners to create accessible, stable, long-lasting resources in the humanities.

Thank you SSHRC!

Social Sciences and Humanities Research Council of Canada

Conseil de recherches en sciences humaines du Canada

Canada

Endings principles

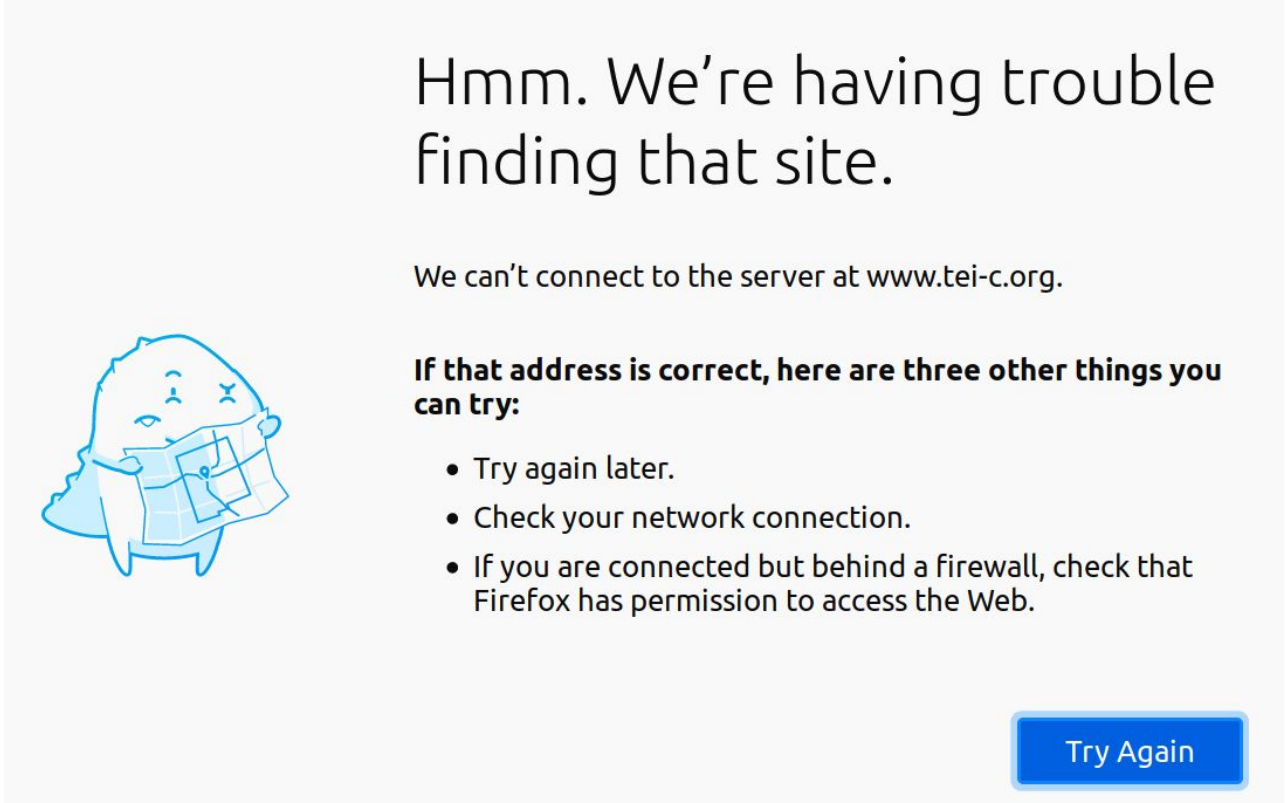
- Data
- Products
- Processing
- Documentation
- Release Management

Endings principles

- Data
- Products
- Processing
- Documentation
- Release Management

Static sites: what and why?

www.tei-c.org, early 2019...



Dependencies

The TEI site depends on WordPress.

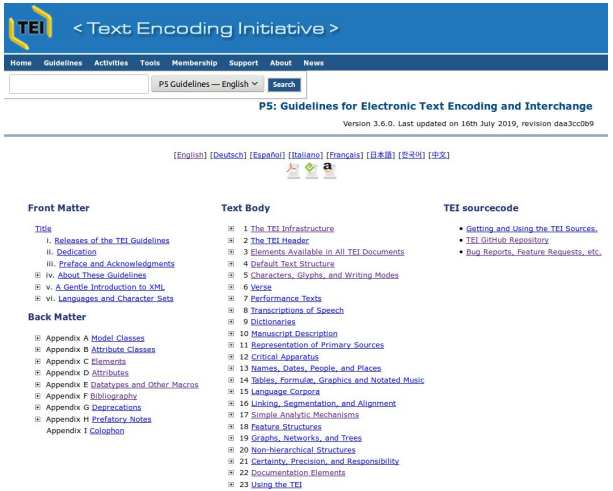
WordPress depends on PHP and MySQL (single point of failure).

Getting a new server running requires

- Installing PHP and MySQL
- Restoring the DB from backup
- Configuring WordPress
- ...

No such problem with the Guidelines

The online version of the TEI Guidelines consists only of HTML, CSS and JavaScript.



It’s always in multiple places.

You can put a copy of it on any server.

You can use it from a local drive.

You can put as many copies as you like wherever you want.

That’s a *static site.*

Static sites:

Have no dependencies (other than a web server) ✅

Are pure HTML, CSS and JavaScript ✅

Can be replicated endlessly ✅

Need little or no curation ✅

Therefore have more chance of survival ✅

We’re not the only people who think so...

Jamstack.org is an initiative to champion “modern web development architecture based on client-side JavaScript, reusable APIs, and prebuilt Markup,” in the interests of “better performance,” “cheaper, easier scaling,” “higher security,” and a “better developer experience.”

MoEML's static site build process: Part 1

- Validate (Relax NG)
- Validate (Schematron)
- Diagnose
    - Coherence: all links point to things.
    - Consistency: all documents conform to encoding and editorial guidelines.
    - Completeness: everything mentioned actually exists.
      (See Holmes and Takeda 2019, http://dx.doi.org/10.1093/llc/fqz011)

If *anything* is wrong, then stop. *A Website with errors is not worth building.*

In the bad old days before Endings...

...project members would happily upload anything they thought was finished into the live web application.

They linked to things that didn’t yet exist or might never exist.

They added images and documents that were never linked to.

The site was always slightly broken in numerous ways.

In the bad old days before Endings...

...project members would happily upload anything they thought was finished into the live web application.

They linked to things that didn't yet exist or might never exist.

They added images and documents that were never linked to.

The site was always slightly broken in numerous ways.

MoEML's static site build process: Part 2

Make more XML!

"Original" TEI: Better versions of our actual encoded files.

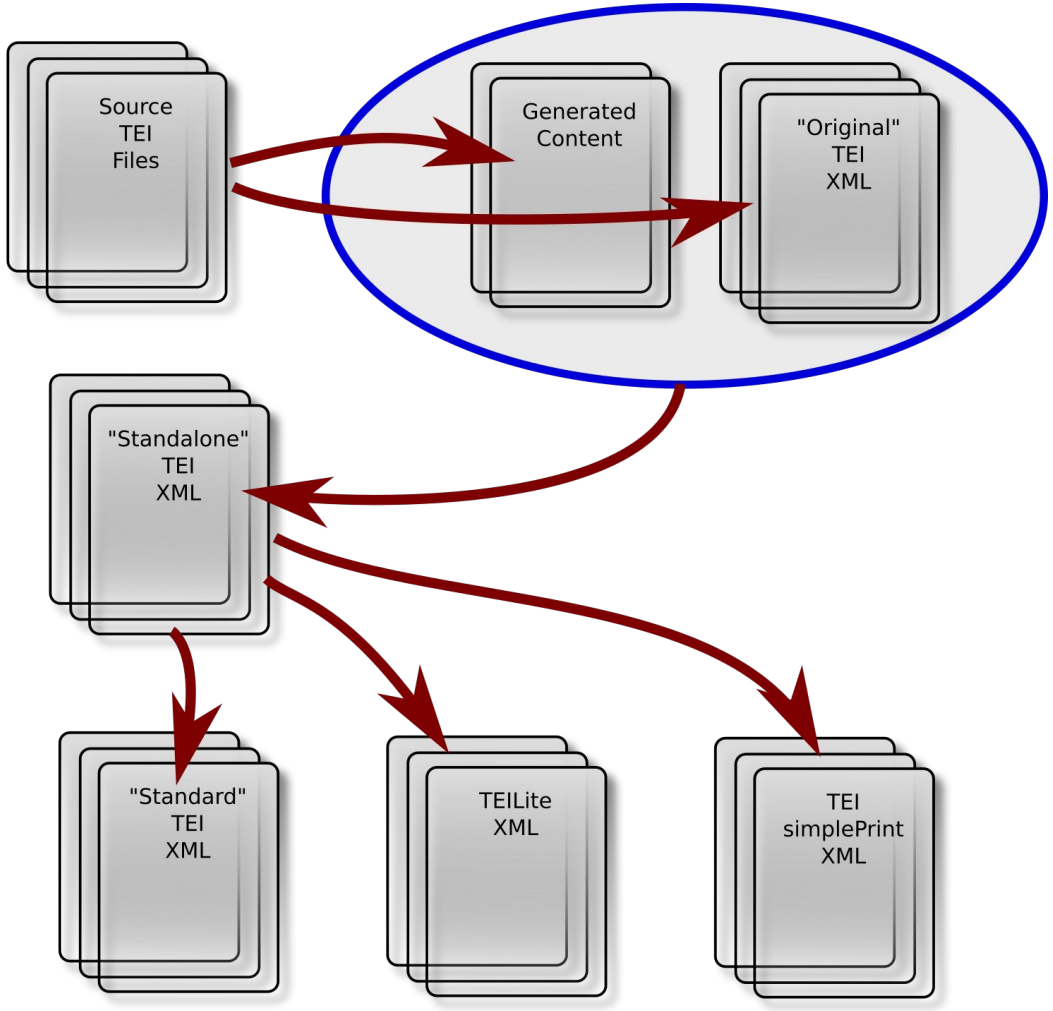
"Synthesized" TEI: New documents computed from existing pages.

"Standalone" TEI: Versions with no external dependencies.

"Standard" TEI: Versions with less common encoding practices normalized.

TEI Lite and TEI simplePrint: Versions for simple interchange.

(See Holmes 2017, http://dx.doi.org/10.1093/llc/fqw048)


Source
TEI
Files
Generated
Content
"Original"
TEI
XML
"Standalone"
TEI
XML
"Standard"
TEI
XML
TEILite
XML
TEI
simplePrint
XML

MoEML’s static site build process: Part 3

- Build the HTML, CSS and JavaScript.
- Validate all the HTML.
- Validate all the CSS.

If *anything* is wrong, then stop. *A Website with errors is not worth building.*

Rules for Website generation

- **Every entity** (location, person, org, article etc.) **has a unique id**, and its own **unique page** on the site.
- **No URL is ever abandoned**. Obsoleted ids still get a page, redirecting as appropriate.
- **Every page stands alone and complete** in terms of content.
- **All pages live together in the same folder**.

“But but but ….”

But massive duplication of content across the site!

We don’t care.

But thousands of files no-one may ever see!

We don’t care.

But 10,591 files all in the same folder!

We don’t care.

Advantages: 1

You can build hugely complex resources that could not be built on the fly.

For example:

- the Gazetteer (9000 entries)
- the A-Z Index (a single page with 10,500 entity ids)

(Because Control + F.)

Advantages: 2

You can do really complicated, multi-pass processing of gnarly markup.

For example: `@style`, `<rendition>/@selector`

- Collect all identical `@style` on elements into a `<rendition>`
- Convert all `<rendition>/@selectors` into XPath, then to `@rendition`
- Then generate HTML/CSS

Advantages: 3

Archiving the site is easy:

```
cp -r site somewhere_else/
```

Replicating the site is easy:

```
cp -r site somewhere_else/
```

Advantages: 4

Everything is there.

Internal links never break.

Everything works.

Everything is valid.

Everyone is happy.

Disadvantages

Deferred gratification.

- Builds take a long time.
- Builds often fail.
- It can be hours before you can see your work on the build server.

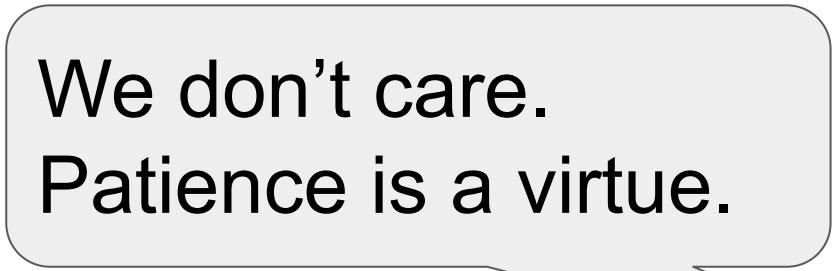
But guess what?

Disadvantages

Deferred gratification.

- Builds take a long time.
- Builds often fail.
- It can be hours before you can see your work on the build server.

But guess what?



Conclusions

Conclusions

- **Everything that can be prefabricated should be prefabricated.**

Conclusions

- **Everything that can be prefabricated should be prefabricated.**
- **Everything that could conceivably be useful should be created and included.**

Conclusions

- **Everything that can be prefabricated should be prefabricated.**
- **Everything that could conceivably be useful should be created and included.**
- **Redundancy is beneficial; in fact it is elegant.**
    - If the same personography entry is replicated in fifty pages that mention that person, then good; any of those pages can now be used outside the context of the collection without loss.

Conclusions

- **Everything that can be prefabricated should be prefabricated.**
- **Everything that could conceivably be useful should be created and included.**
- **Redundancy is beneficial; in fact it is elegant.**
    - If the same personography entry is replicated in fifty pages that mention that person, then good; any of those pages can now be used outside the context of the collection without loss.
- **Patience is a virtue: let your build take a long time; let your releases be well-separated.**

Thanks for listening!

https://mapoflondon.uvic.ca

https://projectendings.github.io/

mholmes@uvic.ca

joey.takeda@gmail.com